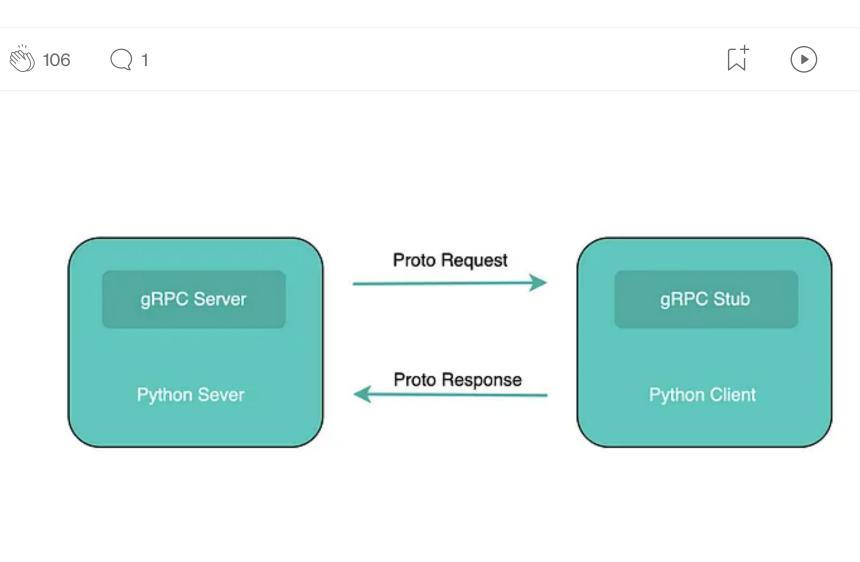
Medium

Q Search Write

2 min read · Dec 29, 2023



Simple Usage of gRPC with Python

Introduction

Barkhayot

Follow

Setting Up Before we dive into the code, make sure you have the necessary tools installed

gRPC (gRPC Remote Procedure Calls) is an open-source framework

using gRPC with Python to create a basic client-server application.

developed by Google that facilitates communication between distributed

Procedure Call) APIs. In this blog post, we'll explore a simple example of

systems by defining and implementing efficient and extensible RPC (Remote

pip install grpcio grpcio-tools

Defining the Service The first step is to define the service using Protocol Buffers (protobuf).

Create a file named calculator.proto with the following content:

```
syntax = "proto3";
package calculator;
```

service Calculator {

int32 num1 = 1;int32 num2 = 2;

```
Generating gRPC Files
Run the following commands to generate the gRPC files:
   python -m grpc_tools.protoc -I. --python_out=. --grpc_python_out=. calculator.pr
This will generate calculator_pb2.py and calculator_pb2_grpc.py.
```

def serve(): server = grpc.server(futures.ThreadPoolExecutor(max_workers=10)) calculator_pb2_grpc.add_CalculatorServicer_to_server(CalculatorServicer(), s

return calculator_pb2.AddResponse(result=result)

class CalculatorServicer(calculator_pb2_grpc.CalculatorServicer):

```
server.start()
        server.wait_for_termination()
    if __name__ == '__main__':
        serve()
Implementing the Client
Create a file named client.py:
    import grpc
    import calculator_pb2
    import calculator_pb2_grpc
    def run(num1, num2):
        with grpc.insecure_channel('localhost:50051') as channel:
            stub = calculator_pb2_grpc.CalculatorStub(channel)
```

In another terminal, run the client:

Running the Server and Client

In one terminal, run the server:

python server.py

python client.py

your projects.

security options to enhance your understanding of this versatile framework. The official gRPC documentation is an excellent resource for diving deeper into advanced topics and best practices.

Source code: https://github.com/barkhayot/grpc-python-example

You should see the result of adding 10 and 20 printed by the client.

simple client-server application. gRPC provides a powerful and efficient way

systems. Explore further to discover more features and use cases for gRPC in

Feel free to experiment with additional gRPC features, error handling, and

to define and implement APIs for communication between distributed

Happy coding!

Grpc

Python

Abraham Ayamigah

Jan 10, 2024 👋 3

Barkhayot

Optimizing ClickHouse

Connections in Go

Backend

```
Responses (1)
     Write a response
 What are your thoughts?
```

Memory Leak? Barkhayot Barkhayot **Building a Flask API Gateway for** How to simply detect and fix gRPC Microservices: A Practical... memory leaks in Go with pprof? In this post, we'll explore the process of Memory leaks can be challenging, even in creating a simple Flask API Gateway to... languages like Go that have automatic...

Nov 2, 2024 **3** 25

Barkhayot

Dec 27, 2023 17

Demystifying gRPC: A

In the realm of modern software

Comprehensive Guide to High-...

development, the quest for efficient and...

Recommended from Medium

SOMPLE

Milad Fahmy

JSON

Data serialization is at the heart of modern software development. Whether you're... K Dec 2, 2024

Using APIs with

Python: SOAP vs

REST vs GraphQL

Protobuf, JSON, and XML: A

Practical Comparison



3d ago

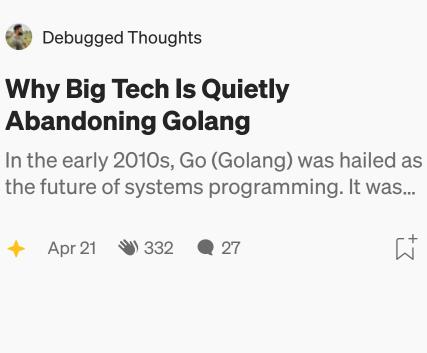
Abhinav

6d ago

In T3CH by Py-Core Python Programming

Using APIs with Python: SOAP vs

Zig Is Quietly Doing What Go Fans Always Wanted



Go, also known as Golang, has been a darling



See more recommendations

of the programming world since its release ...

Sign up

Sign in

rpc Add (AddRequest) returns (AddResponse); message AddRequest {

message AddResponse { int32 result = 1;

Implementing the Server Create a file named server.py:

import calculator_pb2

from concurrent import futures

def Add(self, request, context):

result = request.num1 + request.num2

server.add_insecure_port('[::]:50051')

import calculator_pb2_grpc

import grpc

response = stub.Add(calculator_pb2.AddRequest(num1=num1, num2=num2)) print(f"Result: {response.result}") if __name__ == '__main__': # Get user Input num1 = int(input("Please input num1: ")) num2 = int(input("Please input num2: ")) run(num1, num2)

Conclusion This example demonstrates a basic usage of gRPC with Python for building a

106 Written by Barkhayot **Follow** 7 Followers · 2 Following

Microservices

Distributed Systems

0

 \Box ⁺

L+

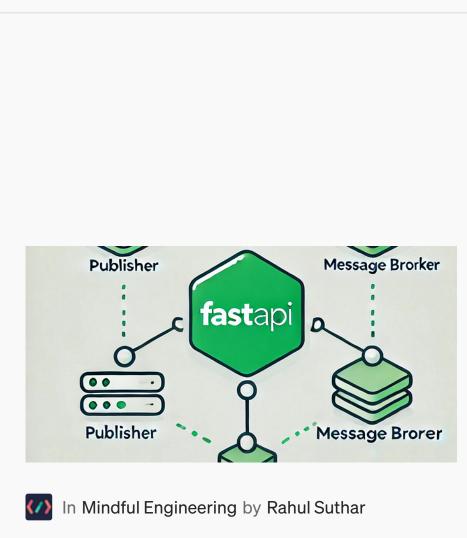
Oct 31, 2024 Good content but it will be great explaining some parts of your codes. <u>Reply</u> More from Barkhayot

ClickHouse is a powerful columnar database management system well-suited for real-tim... 4 Jun 26, 2024 **3** 25 See all from Barkhayot

Optimize Your

Clickhouse

Connections in Go



Understanding Message Brokers

and Message Backends: Redis,...

Dec 17, 2024 33

Now, let's explore Kafka, a powerful message

broker used for high-throughput, low-latenc...

70,0

 \Box ⁺

Why Big Tech Is Quietly **Abandoning Golang**

Leave SaaS lock-in behind with these lightweight, self-hosted alternatives 5d ago 166

K

Help Status About Careers Press Blog Privacy Rules Terms Text to speech